Docket No. 212029US2X

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: Emmanuel FUCHS, et al. GAU:

SERIAL NO: NEW APPLICATION EXAMINER:

FILED: HEREWITH

FOR: UNIVERSAL COMPUTER CODE GENERATOR

REQUEST FOR PRIORITY

ASSISTANT COMMISSIONER FOR PATENTS
WASHINGTON, D.C. 20231

SIR:

☐ Full benefit of the filing date of U.S. Application Serial Number, filed, is claimed pursuant to the provisions of **35 U.S.C. §120**.

☐ Full benefit of the filing date of U.S. Provisional Application Serial Number , filed , is claimed pursuant to the provisions of **35 U.S.C. §119(e)**.

■ Applicants claim any right to priority from any earlier filed applications to which they may be entitled pursuant to the provisions of **35 U.S.C. §119**, as noted below.

In the matter of the above-identified application for patent, notice is hereby given that the applicants claim as priority:

COUNTRY	APPLICATION NUMBER	MONTH/DAY/YEAR
FRANCE	00 09971	July 28, 2000

Certified copies of the corresponding Convention Application(s)

■ are submitted herewith

☐ will be submitted prior to payment of the Final Fee

☐ were filed in prior application Serial No. filed

☐ were submitted to the International Bureau in PCT Application Number .
Receipt of the certified copies by the International Bureau in a timely manner under PCT Rule 17.1(a) has been acknowledged as evidenced by the attached PCT/IB/304.

☐ (A) Application Serial No.(s) were filed in prior application Serial No. filed ; and

(B) Application Serial No.(s)

☐ are submitted herewith

☐ will be submitted prior to payment of the Final Fee

Respectfully Submitted,

OBLON, SPIVAK, McCLELLAND,
MAIER & NEUSTADT, P.C.

Marvin J. Spivak
Registration No. 24,913

22850
Tel. (703) 413-3000
Fax. (703) 413-2220
(OSMMN 10/98)

R E P U B L I Q U E F R A N Ç A I S E

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

BREVET D'INVENTION

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le 05 JUIL. 2001

Pour le Directeur général de l'Institut
national de la propriété industrielle
Le Chef du Département des brevets

Martine PLANCHE

INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE
SIEGE
26 bis, rue de Saint Petersbourg
75800 PARIS cedex 08
Téléphone : 01 53 04 53 04
Télécopie : 01 42 93 59 30
http://www.inpi.fr

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : 01 53 04 53 04 Télécopie : 01 42 94 86 54

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ

Code de la propriété intellectuelle - Livre VI

cerfa N° 11354*01

REQUÊTE EN DÉLIVRANCE 1/2

Cet imprimé est à remplir lisiblement à l'encre noire DB 540 W /260899

Réservé à l'INPI

REMISE DES PIÈCES
DATE 28 JUIL 2000
LIEU 75 INPI PARIS

N° D'ENREGISTREMENT NATIONAL ATTRIBUÉ PAR L'INPI 0009971

DATE DE DÉPÔT ATTRIBUÉE PAR L'INPI 28 JUIL. 2000

V s références pour ce dossier *(facultatif)* 62178

1 NOM ET ADRESSE DU DEMANDEUR OU DU MANDATAIRE À QUI LA CORRESPONDANCE DOIT ÊTRE ADRESSÉE

Laurent LUCAS
THOMSON-CSF
TPI/DB
13, avenue du Président Salvador Allende
94117 ARCUEIL Cédex

Confirmation d'un dépôt par télécopie ☐ N° attribué par l'INPI à la télécopie

2 NATURE DE LA DEMANDE	**Cochez l'une des 4 cases suivantes**	
Demande de brevet	☒	
Demande de certificat d'utilité	☐	
Demande divisionnaire	☐	
Demande de brevet initiale	N°	Date / /
ou demande de certificat d'utilité initiale	N°	Date / /
Transformation d'une demande de brevet européen *Demande de brevet initiale*	☐ N°	Date / /

3 TITRE DE L'INVENTION (200 caractères ou espaces maximum)

GENERATEUR UNIVERSEL DE CODE INFORMATIQUE.

4 DÉCLARATION DE PRIORITÉ OU REQUÊTE DU BÉNÉFICE DE LA DATE DE DÉPÔT D'UNE DEMANDE ANTÉRIEURE FRANÇAISE	
	Pays ou organisation Date / / N°
	Pays ou organisation Date / / N°
	Pays ou organisation Date / / N°
	☐ **S'il y a d'autres priorités, cochez la case et utilisez l'imprimé «Suite»**

5 DEMANDEUR		☐ **S'il y a d'autres demandeurs, cochez la case et utilisez l'imprimé «Suite»**
Nom ou dénomination sociale		AIRSYS ATM S.A.
Prénoms		
Forme juridique		
N° SIREN		. . . . .
Code APE-NAF		. . . .
Adresse	Rue	19, rue de la Fontaine
	Code postal et ville	92221 BAGNEUX
Pays		FRANCE
Nationalité		Française
N° de téléphone *(facultatif)*		
N° de télécopie *(facultatif)*		
Adresse électronique *(facultatif)*		

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ

REQUÊTE EN DÉLIVRANCE 2/2

Réservé à l'INPI

REMISE DES PIÈCES
DATE 28 JUIL 2000
LIEU 75 INPI PARIS

N° D'ENREGISTREMENT NATIONAL ATTRIBUÉ PAR L'INPI 0009971

DB 540 W /260899

Vos références pour ce dossier : *(facultatif)*		62178
6 MANDATAIRE		
Nom		LUCAS
Prénom		Laurent
Cabinet ou Société		THOMSON-CSF TPI/DB
N °de pouvoir permanent et/ou de lien contractuel		
Adresse	Rue	13, Avenue du Président Salvador Allende
	Code postal et ville	94117 ARCUEIL
N° de téléphone *(facultatif)*		01.41.48.45.41
N° de télécopie *(facultatif)*		01.41.48.45.01
Adresse électronique *(facultatif)*		
7 INVENTEUR (S)		
Les inventeurs sont les demandeurs		☐ Oui ☒ Non **Dans ce cas fournir une désignation d'inventeur(s) séparée**
8 RAPPORT DE RECHERCHE		**Uniquement pour une demande de brevet (y compris division et transformation)**
Établissement immédiat ou établissement différé		☒ ☐
Paiement échelonné de la redevance		**Paiement en trois versements, uniquement pour les personnes physiques** ☐ Oui ☒ Non
9 RÉDUCTION DU TAUX DES REDEVANCES		**Uniquement pour les personnes physiques** ☐ Requise pour la première fois pour cette invention *(joindre un avis de non-imposition)* ☐ Requise antérieurement à ce dépôt *(joindre une copie de la décision d'admission pour cette invention ou indiquer sa référence)* :
Si vous avez utilisé l'imprimé «Suite», indiquez le nombre de pages jointes		

10 SIGNATURE DU DEMANDEUR OU DU MANDATAIRE (Nom et qualité du signataire)

Laurent LUCAS

VISA DE LA PRÉFECTURE OU DE L'INPI

La loi n°78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés s'applique aux réponses faites à ce formulaire. Elle garantit un droit d'accès et de rectification pour les données vous concernant auprès de l'INPI.

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ

Code de la propriété intellectuelle - Livre VI

cerfa N° 11 235*02

DÉPARTEMENT DES BREVETS

26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : 01 53 04 53 04 Télécopie : 01 42 93 59 30

DÉSIGNATION D'INVENTEUR(S) Page N° 1../1..

(Si le demandeur n'est pas l'inventeur ou l'unique inventeur)

Cet imprimé est à remplir lisiblement à l'encre noire DB 113 W /260899

V s références pour ce dossier *(facultatif)*	62178
N° D'ENREGISTREMENT NATIONAL	0009971

TITRE DE L'INVENTION (200 caractères ou espaces maximum)

GENERATEUR UNIVERSEL DE CODE INFORMATIQUE.

LE(S) DEMANDEUR(S) :

AIRSYS ATM S.A.

DESIGNE(NT) EN TANT QU'INVENTEUR(S) : (Indiquez en haut à droite «Page N° 1/1» S'il y a plus de trois inventeurs, utilisez un formulaire identique et numérotez chaque page en indiquant le nombre total de pages).

Nom		FUCHS	
Prénoms		Emmanuel	
Adresse	Rue	THOMSON-CSF TPI/DB 13, Avenue du Président Salvador Allende	
	Code postal et ville	94117	ARCUEIL CEDEX
Société d'appartenance *(facultatif)*			
Nom		LACOSTE	
Prénoms		Thierry	
Adresse	Rue	THOMSON-CSF TPI/DB 13, Avenue du Président Salvador Allende	
	Code postal et ville	94117	ARCUEIL CEDEX
Société d'appartenance *(facultatif)*			
Nom			
Prénoms			
Adresse	Rue		
	Code postal et ville		
Société d'appartenance *(facultatif)*			

DATE ET SIGNATURE(S) DU (DES) DEMANDEUR(S) OU DU MANDATAIRE (Nom et qualité du signataire)

28 JUIL. 2000
Laurent LUCAS

ORIGINAL

La présente invention concerne un générateur universel de code informatique. Elle s'applique notamment pour la génération automatique de code telle que par exemple pour des translations de langages informatiques.

Un but d'un générateur automatique de code est notamment de permettre une économie de temps et donc de réduire les coûts de développement des systèmes utilisant des programmes informatiques. En effet, le code généré peut être représenter un nombre très important de lignes, par exemple plusieurs dizaines de milliers de lignes. Ce qui demande beaucoup de temps à un ou plusieurs programmeurs. La génération automatique de code présente par ailleurs d'autres avantages par rapport à une programmation humaine. En effet, le code généré est uniforme. Ainsi, lorsqu'une erreur est détectée, la correction peut être appliquée uniformément dans tous les fichiers comportant cette erreur.

Un générateur de code nécessite en premier lieu un fichier de spécifications. Ce fichier comporte en fait les données provenant de l'utilisateur. Ces données sont présentées par exemple sous un langage informatique évolué, par exemple des langages tels que IDL 2, IDL 3 ou ADA. Le générateur nécessite par ailleurs un ensemble de règles par lesquels l'utilisateur définit l'utilisation des données présentes dans le fichier de spécification afin de définir le code approprié. Un inconvénient est qu'il est nécessaire de définir ces règles en fonction de chaque langage. En d'autres termes, pour chaque nouveau langage, un nouvel ensemble de règles doit être défini. Or il existe une grande quantité de langages. Il faut donc réaliser autant de générateurs de codes qu'il existe de langages informatiques, ce qui est lourd et coûteux.

Enfin, de nombreux systèmes utilisent des interfaces communes programmées dans un même langage. L'hétérogénéité des langages utilisés par ailleurs impose donc de traduire ces derniers en des langages adaptés aux interfaces, c'est-à-dire de créer un code adapté à ces derniers.

Un but de l'invention est de permettre la réalisation d'un générateur de code automatique indépendant des langages utilisés. A cet effet, l'invention a pour objet un générateur de code informatique à partir d'un fichier de spécifications qui comporte au moins :

- une unité frontale FE créant un fichier intermédiaire par analyse grammaticale et syntaxique du fichier de spécifications, le fichier intermédiaire comportant un arbre syntaxique décrivant les données du fichier de spécifications, chaque donnée extraite de ce fichier par l'unité frontale FE étant associée à un nœud de l'arbre ;
- un fichier d'instructions définissant les règles de programmation associées à chaque nœud, en fonction du code à générer ;
- une unité de codage BE générant le code par lecture du fichier intermédiaire et de l'arbre syntaxique.

L'unité frontale FE lit un fichier décrivant la grammaire du langage du fichier de spécifications. Elle décompose le fichier de spécifications en éléments logiciels formant les nœuds de l'arbre syntaxique, selon une arborescence fonctionnelle conforme au fichier de spécifications, les éléments logiciels étant les données extraites de ce fichier.

Le fichier d'instructions comporte les règles de programmation du langage de sortie associées à chaque élément logiciel d'un nœud, une règle et la façon dont est appliquée cette règle étant associée à chaque nœud. Les éléments logiciels associés aux nœuds sont par exemple des interfaces, des variables, des constantes, des opérations et fonctions logiques ou mathématiques.

L'invention a notamment pour principaux avantages qu'elle permet une économie de réalisation d'un générateur de code informatique et qu'elle permet une grande souplesse d'utilisation d'un tel générateur.

D'autres caractéristiques et avantages de l'invention apparaîtront à l'aide de la description qui suit faite en regard de dessins annexés qui représentent :

- la figure 1, une représentation par son architecture logicielle d'un générateur de code selon l'invention ;
- la figure 2, un exemple de représentation codée et un exemple de représentation schématique d'un arbre syntaxique utilisé par un codeur selon l'invention.

La figure 1 illustre un exemple de réalisation d'un générateur de code selon l'invention. Il comporte plusieurs parties dont notamment une unité frontale FE, encore appelée par la suite "Front End", une unité de codage BE, encore appelée par la suite "Back End" et un fichier d'instruction 3, encore appelé par la suite "Template".

L'unité "Front End" FE communique avec le fichier des spécifications 4 d'un langage de départ, par exemple un langage à translater, c'est-à-dire traduire, dans un autre langage. Cette unité frontale FE communique aussi avec un fichier 5 décrivant la grammaire du langage du fichier de spécifications. L'unité frontale FE traite donc notamment avec deux séries de données d'entrée, celle fournies par le fichier de spécifications 4, décrivant notamment la syntaxe du langage, et celles fournies par le fichier de grammaire 5. L'analyse grammaticale effectuée de façon classique par l'unité FE permet à cette dernière d'extraire les données de plus haut niveau contenues dans le fichier de spécifications 4.

L'unité frontale FE génère un fichier intermédiaire 6 qui comporte un arbre syntaxique représentant le fichier analysé 4. Cet arbre contient toutes les données extraites de ce fichier de spécifications 4 mais ne conserve aucune des spécificités du langage utilisé dans ce fichier 4. Le fichier intermédiaire 6 est par exemple écrit en code ASCII.

L'unité "Back End" BE délivre le fichier de spécifications de sortie 7. Ce fichier comporte par exemple la traduction du langage d'entrée 4 dans un nouveau langage informatique. Le langage de sortie peut être tout

type de langage, tel que par exemple le langage C ou le langage ADA. L'unité BE nécessite deux séries de données d'entrée, les données fournies par le fichier intermédiaire 6 créé par l'unité frontale FE, c'est-à-dire l'arbre syntaxique qu'il contient, et les données fournies par le fichier "Template" 3. Ce dernier décrit la façon d'utiliser les données contenues dans le fichier intermédiaire 6 et comment les traiter pour générer le bon code. Avantageusement, l'unité BE est entièrement générique et n'est jamais changée ou modifiée lorsqu'un nouveau fichier de spécifications 4, donc un nouveau langage, est présent à l'entrée de l'unité frontale FE. Le fichier "Template" 3 doit être créé par les utilisateurs. Ce fichier est notamment fonction du code ou langage à générer. Il permet notamment d'accéder aux données venant des spécifications d'entrée 4, de les manipuler et de les transformer, et ainsi permet à l'unité BE de générer le code attendu. A cet effet, il comporte notamment toutes les fonctionnalités nécessaires pour extraire les données du fichier intermédiaire 6.

La figure 2 présente un exemple d'organisation du fichier intermédiaire 6. L'unité frontale FE décompose les spécifications d'entrée 4 en éléments logiciels de base 21, 22, 23, 24, 25, 26 formant des nœuds reliés entre eux selon une arborescence fonctionnelle conforme aux spécifications 4. Les éléments logiciels sont les données extraites du fichier de spécifications 4. A titre d'exemple l'arbre syntaxique illustré par la figure 2 comporte un nombre réduit d'unités logicielles, dans le but notamment de faciliter la description du fichier intermédiaire 6. Cet arbre syntaxique est représenté sous un exemple de forme codée 20 et sous un exemple de forme schématique 30. En particulier, l'arbre syntaxique de la figure 2 se rapporte à un sous-programme, c'est-à-dire une structure ou un module m, contenu dans le fichier de spécifications 4. Par suite de l'analyse grammaticale et syntaxique de ce fichier 4, l'unité frontale FE détecte un module m. Elle crée alors un nœud 21 correspondant à ce module, ce nœud étant par exemple intégré dans une arborescence plus générale. Puis, l'analyse effectuée par l'unité FE lui permet de détecter que le module m traite deux interfaces, une interface i_1 et une interface i_2. En conséquence, elle crée un nœud 22 correspondant à l'interface i_1 relié en aval au nœud 21 du module m, et elle crée un nœud 23 correspondant à l'interface i_2 relié en

aval au nœud 21 du module m. En descendant d'un niveau dans l'arborescence, un premier nœud 24 et un deuxième nœud 25 sont reliés au nœud 22 de l'interface i_1, et un troisième nœud 26 est relié au nœud 23 de l'interface i_2. Les premier et deuxième nœuds 24, 25 correspondent respectivement à une variable et à une constante utilisées pour l'interface i_1. Le troisième nœud correspond à une constante c utilisée pour l'interface i_2. Les nœuds associés aux variables et les constantes peuvent notamment être placés en bout de branche. Dans le cas d'une application de gestion de trafic aérien, ces variables sont par exemple des données relatives aux avions ou aux pistes.

Un programmateur du fichier d'instruction 3 doit notamment manipuler les données contenues dans le fichier intermédiaire 6. Or, ces données peuvent être stockées de telle sorte qu'elles sont difficilement accessibles. A cet effet, l'arbre syntaxique peut être présenté sous forme schématique 30 au programmateur. C'est en fait une présentation des données de spécifications originales 4 organisées selon un arbre logique. Ainsi, pour chaque structure du langage de spécifications utilisé, le fichier d'instructions 3 permet d'accéder aux données concernant cette structure, par exemple le module m, les interfaces i_1, i_2, les constantes s et a, et de parcourir cette structure dans l'arbre syntaxique.

Dans le fichier d'instructions 3, à chaque nœud de l'arbre syntaxique est associée une règle et la façon dont est appliquée cette règle. L'arbre syntaxique permet par ailleurs de reconnaître la nature des objets ou éléments logiciels associés aux nœuds. Les natures possibles sont par exemple les interfaces, les variables, les constantes ou encore les opérations et fonctions logiques ou mathématiques. Les règles sont adaptées à chaque type ou chaque nature d'élément logiciel. En particulier, lorsqu'il s'agit d'une interface, l'adresse physique de cette dernière doit être indiquée ainsi que par exemple un protocole de communication connu par ailleurs. Les variables et les constantes doivent être précisées. Il peut par exemple s'agir de variables représentatives de vitesses, de position, ou encore de n'importe quel types de grandeurs physiques. A titre d'exemple, pour le nœud 21 associé au module m, on indique dans le fichier d'instructions 3 les règles de

programmation des interfaces i_1 et i_2. De même pour chacun des nœuds 22, 23 associés à ces interfaces, on indique les règles de programmation des constantes ou des variables correspondant aux nœuds de niveau inférieur 24, 25, 26. Les règles de programmation associées à chaque nœud sont bien sûr fonction du langage de sortie 7. Le fichier d'instruction comporte donc les règles de programmation du langage de sortie, associées à chaque élément logiciel d'un nœud.

Le fichier d'instructions 3 commande en quelque sorte l'unité BE. Il permet notamment d'extraire les données 21, 22, 23, 24, 25, 26 du fichier intermédiaire 6 et de commander la génération de code effectuée par l'unité BE qui utilise ces données. Il est en particulier possible de créer autant de fichiers d'instructions 3 qu'il est nécessaire pour traiter un même type de fichier de spécifications d'entrée. Ce même fichier d'entrée peut alors être la source de plusieurs générations de codes différentes, c'est-à-dire en fait de plusieurs langages différents. En particulier, un fichier d'instructions 3 est associé à chaque langage de sortie.

Plusieurs types de langage de programmation peuvent être utilisés pour créer le fichier d'instruction 3. Un langage particulier peut être adapté à l'écriture de ce fichier 3. Ce langage sera appelé langage TDL par la suite, selon l'expression anglo-saxonne « Templates Description Language ». Le langage TDL peut être un langage de programmation complet qui traite tous les éléments logiciels classiques tels que par exemples ceux que le langage C peut permettre, c'est-à-dire notamment les contrôles de structure (if, for, do, while...), les variables, les fonctions et les classes. Sa syntaxe peut ainsi par exemple ressembler à celle du langage C^{++} permettant à un programmeur de créer facilement un fichier d'instructions 3. Le langage TDL permet de manipuler facilement toutes sortes de variables ou d'objets. Il peut être puissant pour manipuler des chaînes de données ou d'instructions. Le code généré 7 qui n'est rien d'autre qu'une concaténation d'une multitude de chaînes d'instructions peut ainsi être produit facilement. Le langage TDL comporte par exemple toutes les instructions nécessaires pour parcourir l'arbre syntaxique 20, 30. En particulier, TDL comporte une classe d'instructions correspondante à chaque

type de nœud de l'arbre permettant notamment d'obtenir toutes les informations que le nœud contient. TDL peut par exemple permettre de mixer du code constant et du code variable. Cela signifie notamment que quelques parties du fichier d'instructions 3 peuvent être reproduites directement dans le fichier de sortie 7 sans interprétation, alors que d'autres parties sont interprétées par l'unité BE. Il est ainsi facile de générer de grandes sections de code constant en plaçant ces sections entièrement dans le fichier d'instructions 3.

Un générateur de code selon l'invention est réellement générique car l'unité de codage BE reste la même quel que soit le langage des spécifications d'entrée 4. En particulier, l'unité BE peut être codée en dur. Les avantages apportés par l'invention sont donc notamment une économie de réalisation et une grande souplesse. L'invention permet ainsi de translater n'importe quel langage informatique, défini par le fichier de spécification d'entrée 4 en n'importe quel autre langage fourni par le fichier de sortie 7. En variante de réalisation, il serait possible d'associer à l'unité frontale FE un fichier d'instructions « Template », ce qui pourrait permettre ainsi de ne pas modifier cette unité en fonction du langage d'entrée.

REVENDICATIONS

1. Générateur de code informatique à partir d'un fichier de spécifications, caractérisé en ce qu'il comporte au moins :

- une unité frontale (FE) créant un fichier intermédiaire (6) par analyse grammaticale et syntaxique du fichier de spécifications (4), le fichier intermédiaire comportant un arbre syntaxique (20, 30) décrivant les données du fichier de spécifications (4), chaque donnée extraite de ce fichier (4) par l'unité frontale (FE) étant associée à un nœud (21, 22, 23, 24, 25, 26) de l'arbre ;
- un fichier d'instructions (3) définissant les règles de programmation associées à chaque nœud, en fonction du code à générer (7) ;
- une unité de codage (BE) générant le code de sortie (7) par lecture du fichier intermédiaire (6) et de l'arbre syntaxique.

2. Générateur selon la revendication 1, caractérisé en ce que l'unité frontale (BE) lit un fichier (5) décrivant la grammaire du langage du fichier de spécifications (4).

3. Générateur selon l'une quelconque des revendications précédentes, caractérisé en ce que l'unité frontale (FE) décompose le fichier de spécifications (4) en éléments logiciels (21, 22, 23, 24, 25, 26) formant les nœuds de l'arbre syntaxique, selon une arborescence fonctionnelle conforme au fichier de spécifications (4), les éléments logiciels étant les données extraites de ce fichier (4).

4. Générateur selon l'une quelconque des revendications précédentes, caractérisé en ce que le fichier d'instructions (3) comporte les règles de programmation du code de sortie (7) associées à chaque élément logiciel d'un nœud, une règle et la façon dont est appliquée cette règle étant associée à chaque nœud.

5. Générateur selon l'une quelconque des revendications précédentes, caractérisé en ce que les éléments logiciels associés aux nœuds sont des interfaces, des variables, des constantes, des opérations et fonctions logiques ou mathématiques.

6. Générateur selon l'une quelconque des revendications précédentes, caractérisé en ce que le code de sortie (7) est un langage informatique.

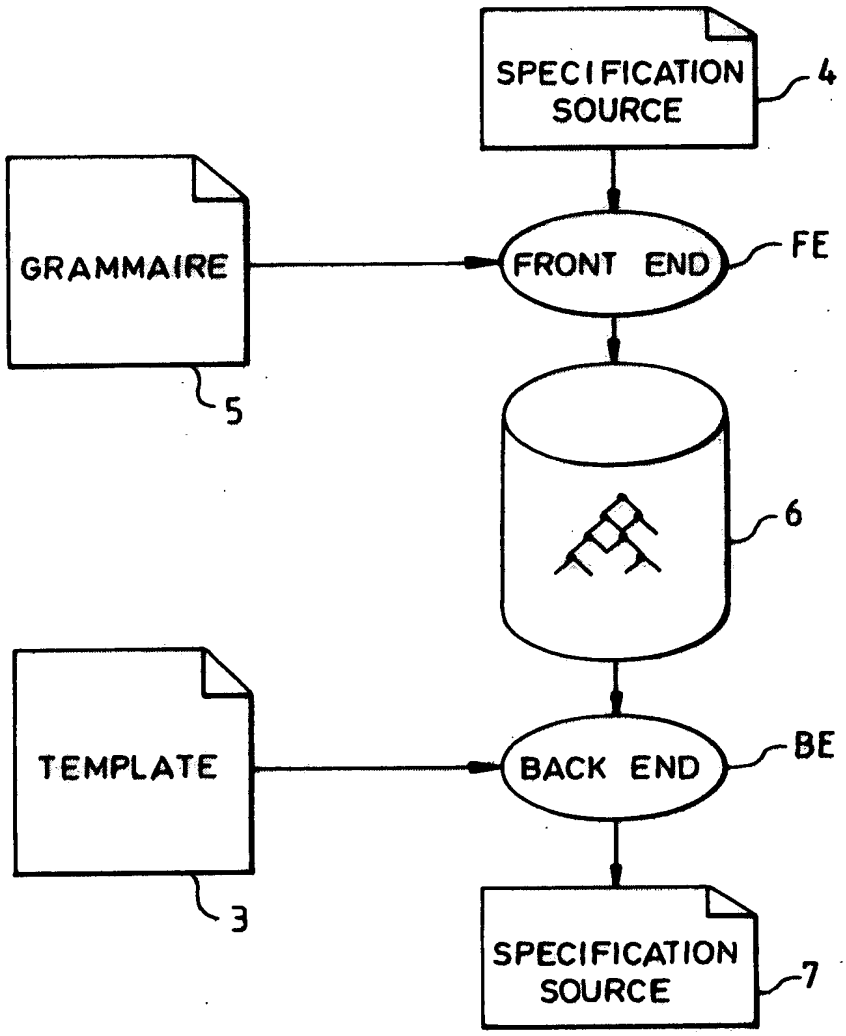
SPECIFICATION SOURCE
4
GRAMMAIRE
FRONT END
FE
5
6
TEMPLATE
BACK END
BE
3
SPECIFICATION SOURCE
7

FIG.1

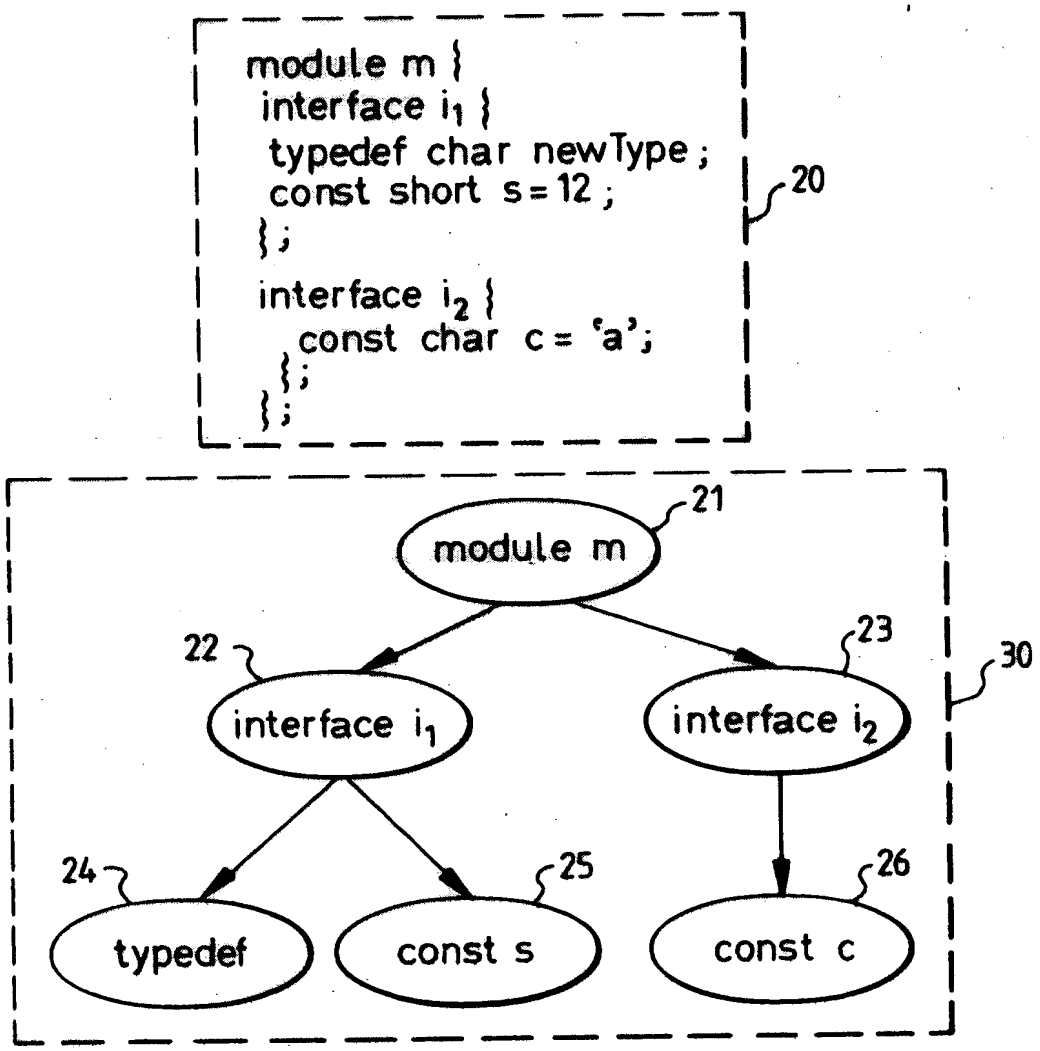
module m {
interface i1 {
typedef char newType;
const short s=12;
};
interface i2 {
const char c='a';
};
};
20
21
module m
22
23
30
interface i1
interface i2
24
25
26
typedef
const s
const c

FIG.2